VZ-LINK NEWSLETTER

<< AUCKLAND VZ 300/200 USERS CLUB >>

ISSUE # 23 AUGUST 1988

Well,here is the August issue of LINK for all you VZ users out there.
I hope you are all working on programs and items for this, your newsletter.Make the most of the wet weather and try some basic programming.

July saw yet another add-on project for the VZ appear in ETI magazine.This time for a DataLogger for use with the VZ.
I hope most of you can make the Vz meeting and workshop on the 27Th.We should have a action packed day.

Peter Paterson will be showing us the Laserlink Eprom's he has just installed.
We also have a couple of other items which will interested you.

There are still a couple of members who haven't renewed their sub's.
If you don't want the newsletter any more,please let me know.

The International correspondence school now use VZ 300's for their home computer courses.I am hoping to arrange for a club info form to go out with these computers.

If any of you have a idea for a competition,please let me know.
It doesn't have to be to hard.
But it must be VZ based and be suitable for all ages.

In this issue we complete the very fine VZ assembly item .If you have problems with assembly or in fact any programs,write to me and we will put a problems page in the newsletter.

Also in this issue we bring the Newsletter index up to date.
In future I will print a index after every five or six issues.

We also have heap's of other item's for all of you.
For details of the user meeting see the next page.

Well,until next issue or I see you at the meeting,** best wishes..

USER MEETING 27 TH AUGUST

CHEQUE BOOK PROGRAM

WRITTEN BY SANDIE MCSWEENEY

```
10 REM SAVINGS ACCOUNT BALANCE
20 REM SANDIE - SU6S2
30 REM T1 TOTAL DEPOSITS ACCUMULATOR
40 REM T2 TOTAL WITHDRAWALS ACCUMULATOR
50 REM D$ MONTH END DATE 31/07/87
60 REM O OLD BALANCE
70 REM D DEPOSIT
80 REM W WITHDRAWAL
90 REM N NEW BALANCE
100 CLS
120 PRINT"ENTER DATE";
130 INPUT D$
140 PRINT"ENTER OLD BALANCE";
150 INPUT O
170 PRINT"ENTER A DEPOSIT OR O";
180 INPUT D
190 PRINT"ENTER A WITHDRAWAL OR O";
200 INPUT W
210 IF D=O AND W=O THEN GOTO 250
220 LET T1=T1+D
230 LET T2=T2+W
240 GOTO 170
250 CLS
260 PRINT"MONTH ENDING:";TAB(23);D$
270 PRINT"OLD BALANCE:";TAB(22);
275 PRINT USING "$$##.##";O
280 PRINT"TOTAL DEPOSITS:";TAB(22);
285 PRINT USING "$$##.##";T1
290 PRINT"TOTAL WITHDRAWALS:";TAB(22);
295 PRINT USING "$$##.##";T2
300 LET N=O+T1-T2
310 PRINT"NEW BALANCE:";TAB(22);
315 PRINT USING "$$##.##";N
320 END
```

VZ USERS-WORKSHOP/MEETING

WE WILL BE HOLDING A VZ USERS WORKSHOP ON:-

SATURDAY 27TH AUGUST 1988
STARTING AT 10-15 AM

AT THE :-

OWAIRAKA BAPTIST CHURCH
119 RICHARDSON ROAD
OWAIRAKA

BRING YOUR FAVOURITE SOFTWARE ETC
BRING YOUR PROBLEM'S

A LIGHT LUNCH WILL BE PROVIDED,BUT PLEASE BRING A PLATE TO HEL

Z80 ASSEMBLY

In this the final part in this series on Assembly Language programming, we look at a simple routine, and at how to store your routines in memory, execute them etc. Imagine that we are required to write a small assembly routine which scans through a given range of memory, looking for a sequence of three values. We may be looking for a certain machine code instruction in ROM, or perhaps a three letter word in a long ASCII string. As with any programming task, a strategy must be devised. This can be in the form of ordinary english sentences. For our particular case, it may read something like this: "Start with the first memory address. READ its contents, and COMPARE this with the first byte in the sequence. IF the two are equal, THEN move to the next memory location and repeat the process, checking for the second and third bytes. OTHERWISE move to the next memory location, COMPARE it with the upper limit of the range to be scanned, and IF this limit is reached, EXIT the routine. IF NOT, repeat the process from the start."

Although this description of the routine operation is not strictly correct, let's assume it is and write a program for it. Notice that certain words in the description have been highlighted, since they represent basic instructions, such as COMPARE (CP), or conditional statements using IF-THEN format. This helps to remind us of what types of instructions will be necessary at various points in the program.

A first attempt at the routine might look like program A (ignore program B for now):

A:

```
001 NUM1 EQU  029H
002 NUM2 EQU  019H
003 NUM3 EQU  029H
004      LD   HL,0FFFFH
005 LOOP INC  HL
006      LD   A,H
007      CP   40H
008      JP   Z,7B00H
009      LD   A,(HL)
010 ;1ST BYTE
011      CP   NUM1
012      JR   NZ,LOOP
013      INC  HL
014      LD   A,(HL)
015 ;2ND BYTE
016      CP   NUM2
017      JR   NZ,LOOP
018      INC  HL
019      LD   A,(HL)
020 ;3RD BYTE
021      CP   NUM3
022      JR   NZ,LOOP
023 ;DISPLAY ADDRESS
024      DEC  HL
025      DEC  HL
026      PUSH HL
027      CALL 0FAFH
028      LD   A,0DH
029      CALL 033AH
030      POP  HL
031      JR   LOOP
```

B:

```
001 NUM1 EQU  029H
002 NUM2 EQU  019H
003 NUM3 EQU  29H
004      LD,  HL,0FFFFH
005 LOOP INC  HL
006      LD   A,H
007      CP   40H
008      JP   Z,7B00H
009      LD   A,(HL)
010 ;1ST BYTE
011      CP   NUM1
012      JR   NZ,LOOP
013      INC  HL
014      LD   A,(HL)
015 ;2ND BYTE
016      CP   NUM2
017      JR   Z,P1
018      DEC  HL
019      JR   LOOP
020 P1   INC  HL
021      LD   A,(HL)
022 ;3RD BYTE
023      CP   NUM3
024      JR   Z,P2
025      DEC  HL
026      DEC  HL
027      JR   LOOP
028 ;DISPLAY ADDRESS
029 P2   DEC  HL
030      DEC  HL
031      PUSH HL
032      CALL 0FAFH
033      LD   A,0DH
034      CALL 033AH
035      POP  HL
036      JR   LOOP
```

Although this routine is not completely correct, we'll look at its operation in detail, and then locate the errors. The first three lines nominate the sequence of values which the routine will look for. Although these could have been scattered throughout the program, they are placed here for convenience and neatness. Note the use of the EQU instruction. This is not a true Z80 instruction, but rather a 'psuedo op-code' which is recognised by the Editor Assembler. These three EQU's make the Ed.Asm. record the three values as NUM1, NUM2 and NUM3. These labels can then be used later in the program, and the Ed.Asm will automatically insert the values in the object code.

This particular routine is designed to scan the bottom 16K of memory, i.e. the ROM. In the next line, HL is set to FFFFH, which is of course the largest value it can hold. The next line (which marks the start of the LOOP routine) increments HL, thus setting it to 0000! If we look back at our description strategy, we have satisfied the first requirement - HL now points to the first memory location to be scanned. Since this part of the routine is a loop we must now check to see that the current memory location is not the end of our range (i.e. 4000H). A shorthand way of doing this is to simply examine the high byte of HL, that is, the H register. If this is 40H, then HL must be greater than or equal to 4000H. Lines 6-8 do this check. 'A' is loaded with 'H', and then compared to 40H. If a match is found, the routine is exited by a jump to the Ed.Asm. (if you are not using the DSE Ed.Asm. you must change this) at 7B00H. This is a very handy exit for any program that is going to be executed FROM WITHIN THE EDITOR. It returns operation to the Command mode of the editor. If the Ed.Asm. is not going to be in memory when the routine is run, then this jump can not be used.

Moving right along in our strategy, we need to next examine the contents of the current memory location stored in HL. This is done in line 9, the value being stored in 'A'. This must be compared to the first number in the sequence with a CP NUM1. If the two do not match, a jump relative takes us back to LOOP to move on to the next location. If they do match, HL is incremented, and the second value looked for. Once again, if the values are not equal, we return to LOOP to start all over again, otherwise the third byte is checked. If a match is found here, then obviously all three numbers have been found in sequence, and it is time to display the location of this sequence. HL is decremented twice so as to point to the first byte of the three. This address is stored on the stack, and then a CALL to 0FAFH is made. This is another useful ROM routine which displays the contents of HL at the current cursor position on the screen. Lines 28-29 simply send a carraige return to the video in order that further displays will be under each other. HL is then POPed off the stack, and the routine goes back to LOOP and continues its search.

While this routine may at first appear 'water-tight', we've chosen it just to illustrate how subtle some errors can be. Before going on, think about the routine yourself - can you see a situation where it would overlook a correct sequence?

Just to prove it to yourself, type in the routine, and assemble & run it. You'll see that only one address, 7788 is given. The sequence being searched for is 29H, 19H and 29H (41, 25, 41 in decimal). Now reset your VZ and PEEK locations 3907, 3908 and 3909. You should find that these locations contain the correct series! So why did the routine fail to detect them?

The reason becomes more obvious when we have a look at

Location 3906,which also contains a 41 (29H),the first number in the sequence.Whenever a routine behaves differently to what you are expecting,follow its process through in your mind. Suppose the routine is just starting the LOOP routine again at Line 5.Since the problem occured around location 3907,let's suppose that HL contains say 3905.At line 5,HL is incremented, making it 3906.The contents of this address,which we know to be 29H,is compared with 29H,and so HL is once again incremented, and the contents of this new address (3907) compared with NUM2=19H. A match is of course not found,since this address contains 29H,so the routine returns to LOOP.It is at this point that the 'bug'occurs.Although HL currently contains 3907 (which is the start of a valid series) it is incremented AGAIN,thus skipping over the start of a correct sequence and missing it entirely. You should be able to see that this occured because of the fact that the first value in the series was doubled up in memory.

Routine B overcomes this problem, by DECrementing HL after a second or third check has been made and failed.

Although this is a very simple routine, it should demonstrate that even apparently simple programs can have hidden bugs in them, which are often very hard to detect. For this reason it is important that you think about your initial strategy very carefully. There are a couple of bugs still in routine B - see if you can find them. Here are some ways you can speed up the job of 'debugging' a program:

1. Draw up a table with line numbers written down the left column, and columns for each of the registers (and perhaps for the flags). Work through the routine mentally, writing the values which each of the registers will assume at each line in the table. Keep another column for the stack, writing each successive PUSH below the previous, and crossing off the bottom entry for each POP (don't forget that a CALL places an address on the stack, and a RET pops one off!)

2. Place breakpoints just before or after the spot where you suspect trouble. This can be done by temporarily inserting a jump to some other routine which displays the contents of the registers (some commercial monitors offer this facility). Even leaving out an instruction might remove the problem.

3. Use structured programming techniques - keep particular jobs in separate subroutines which are labelled appropriately.

Now that you've got your program running, the next question is where to put it! This depends very much on the function of the program, its size, and how much memory is available.

Firstly, a program that is entirely assembly language, and which runs alone (i.e. it is not called from BASIC, and the BASIC command mode is never entered) can be located at the normal start of BASIC, that being 7AE9H = 31465. Simply save your source file, set your origin using '031465' (for the DSE Ed.Asm.) and then save your object code. If you're using a tape version, this object code will run as soon as it is loaded. If you're using a disk based Editor, you should BRUN your routine to run it.

In the majority of cases, however, your assembly language routine will be designed to be called from BASIC using the X=USR(0) statement. In this case, you may wish to embed the code

within or after your BASIC program otherwise, the best option (especially for disk users) is to load the assembly routine first at the top of memory,and set the USR pointers (30862/3) to its starting addresss,then load & run your BASIC program.If you intend adopting this approach,you'll have to classify your routine as either relocatable or non-relocatable.

Put simply,your routine will not be relocatable if it has any JPs or CALLs to subroutines within itself (which includes the majority). JRs are OK.

If your routine fits into this category,you'll have to work out how long it is,and subtract this length from your usual top of memory (don't forget about DOS if you use it). This will be the origin of your routine when you assemble it.Don't forget to reset your top of memory pointer and set aside string space before loading your routine.

If your routine is relocatable,you can use the following header,which I've borrowed from Larry Taylor,to automatically copy your routine into the top of memory,and reset all pointers according.Place it at the start of your program.Your original must start with a 'STRT' label and end with a single byte instruction (such as a NOP) labelled 'END'.Set your origin to 31465. On loading the program will position itself at the top of memory,and do a new,returning to BASIC.The beauty of the header is that it works for whatever memory is available,and thus for whatever machine is being used.Here's the HEADER:

```
001   LD   SP,7700H
002   LD   HL,(78B1H)
003   LD   BC,END-STRT
004   INC  BC
005   PUSH BC
006   XOR  A
007   SBC  HL,BC
008   LD   (78B1H),HL
009   XOR  A
010   LD   BC,33H
011   SBC  HL,BC
012   LD   (78A0H),HL
013   LD   HL,STRT
014   POP  BC
015   LDIR
016   CALL 1B4DH
017   JP   1A19H
```

Thats it for this series.Although there is lots more to talk about,I trust this series has been a useful introduction to assembly language programming.

I would like to thank Mark Harwood for giving permission for this series to be reprinted from his VZ USER newsletter.

We will have more on assembly language in forthcoming issues, But for now try some of the exercises from this very fine series...regards PETER.

INSTALLING A 32K STATIC RAM IN A VZ 200

WRITTEN BY JOHN MASON

WARNING You should be experienced to try this mod.
If your not sure,DON'T

Turn your VZ upside down and Undo the screws and remove the lid.Spring the keyboard loose from the lid.
Next remove the circuit board from the case.

CAREFULLY unsolder the metal shield covering the 6K Ram,which can be seen mounted above the circuit board.
DO NOT remove the circuit board.It consists of three 6116 static Rams and a 74 L.S 138.
The 6116 nearest to the L.S 138 is Ram No1.This is the Ram selected from the main board by 78xx.The other two 6116's will be permanently de-selected by pin 14 of L.S 138 which will now be modified to chip select the 32K Ram,thus giving a 34K memory.

TO MODIFY THE L.S 138

Cut the printed circuit close to pin 1 and pin 2.
Pin 3 is wired through a loop of wire on the underside of the board to a point just above.
Cut through the printed circuit close to this point,Check with a Continuity meter that pin 1,2 and 3 are not connected or shorted to anything on the board.
Join pins 1,2 and 3 together and connect them to the same earth.

Cut the ciruit board at pin 5 to disconnect pin 5.Join pin 5 to pin 4.

Next you will notice that pin 15 and pin 14 are jointed to 2 points on the edge of the board by tinned wire,on the underside.These are the select lines for Ram 2 and 3.
Look underneath,and slip a thin screwdriver under the tinned wire that goes to pin 15.Touch the soldering iron on the point,and gently disconnect the wire that goes to pin 15.It can be removed with a pair of long-nosed pliers.

This leaves pin 15 free to select the 32K Ram.Join the 2 select lines of Ram 2 and 3 together,which will connect them both to pin 14.
As pin 14 is permanently on,this will de-select Ram 2 and 3.
Pin 15 is now used to select the 32K Ram which will come on at Hex 8000, and turn off at Hex FFFF.

PART'S LIST

One P.D 43256C 32K Ram
One low profile socket
One piece of vero board-standard I.C pitch
(34 Metal strips wide and 13 Holes high)
One .1 Disc ceramic capacitor.
Some fine hook-up wire
One thin solder lug.

CONTINUED OVER THE PAGE.

Mount the socket in the centre of the vero board.
This should leave 3 unused strips each end,plus enough either side to solder wires to.
The I.C board will be mounted with the I.C underneath so that one edge will be sitting on top of the 6K board over the etched serial number.
It will be sitting above and partly covering the Z80 chip.

Remove the screw from the stand off installator,slip on the solder lug and screw down.This should put the solder lug over the 3 unused strips on the I.c board.
Solder a wire from the three strips on the other end of the board to the earth plate which is the part with the etched number on.

Pin 28 V.C.C should be down in the corner nearest the solder lug.DON'T insert the I.C at this stage.
Now start to wire the pin's,looping round to the end of the 6K board.All the required signals can be found on the two rows of Pins that connect to the main board.

Trace with a meter using the Pins of a 6116.
E.g. Pin 1 of 6116 is A7.Find A7 pin on the connector end and wire to A7 of the 32K Ram.etc.
Carry on until all of the pins are connected .

REMEMBER Pin 15 of the L.S 138 is connected to chip select of the 32K Ram.
When all the wiring is finished connect the .1 Disc between Pin 28 and Earth.
CHECK the wiring thoroughly.Loosen the solder lug and bend the board up vertical.

Carefully insert the Ram,pressing home well.
Now carefully fold the board horizontal.If the Ram is not fully inserted it will not lie flat,because it hit's on the Z80 chip.

If satisfied hold your breath and switch on.The usual message should appear on the screen.
Type in the usual memory available program,which is:

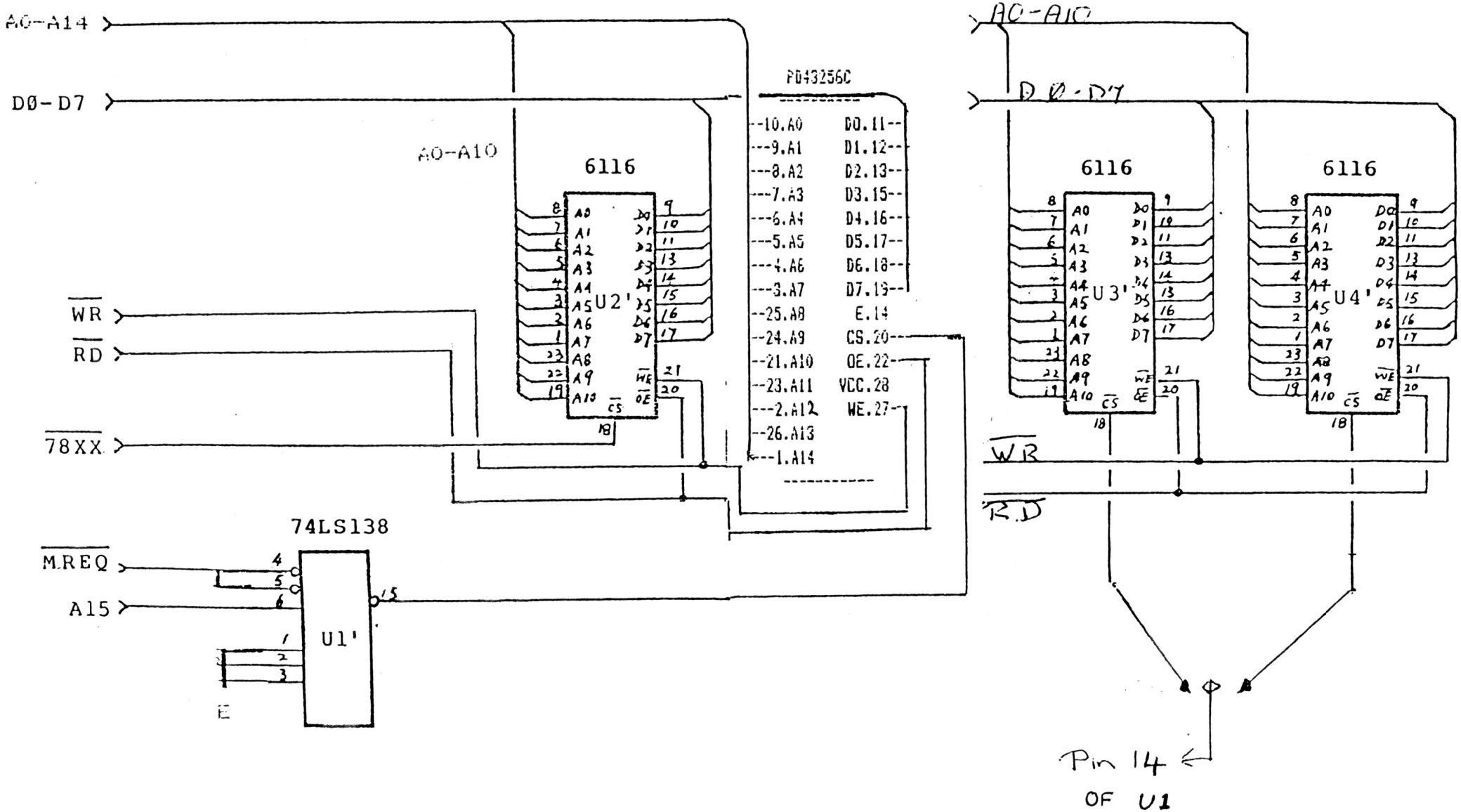
```
printpeek(30897)+256*peek(30898 )
```

and if all is well you should get 65535.

Solder a piece of tinned wire from end of the strips to main board and replace metal cover.

The 32K Static Ram IC PD43256C is available from:
The Microcomputer Electronic co ltd
P.O box 9224, New Market
Auckland

LASER 34K INTERNAL PROGRAM RAM --SCHEMATIC



PART RESTORE DATA

There may be times when a program is not required to READ from the start of the DATA statement as the program runs. This can be achieved using PEEK to read the address stored in the DATA Pointer (which resides at 30975/6 dec. 78FF/7900H) and then POKE that address into the Pointer when the remaining DATA is required. The following program will demonstrate:

```
100 REM - PART RESTORE DATA
110 DATA A,B,C,D,E,F,G,H,I,J
200 CLS:PRINT @ 226,"PART 1";
210 PRINT TAB(20);:FOR X=1 TO 5
220 READ A$:PRINT A$;:NEXT
230 DA=PEEK(30975):DB=PEEK(30976)
300 PRINT @ 258,"PART 2";
310 PRINT TAB(20);:FOR X=1 TO 5
320 READ A$:PRINT A$;:NEXT
400 POKE 30975,DA:POKE 30976,DB
410 PRINT @ 290,"PART 2 - RESTORED";
420 PRINT TAB(20);:FOR X=1 TO 5
430 READ A$:PRINT A$;:NEXT
440 END
```

SCROLL CONTROL

Have you ever been frustrated with a program display which scrolls on before you can react in time to stop it? The problem can be controlled by using PEEK to read the position stored in the Cursor Pointer (30752/3 dec. 7820/1H) and compare it with a predetermined position (usually the last line of the screen).

```
100 REM SCROLL CONTROL
110 CLS
120 X=0
130 PRINT @ X,X
140 X=X+32
150 C=PEEK(30752)+256*PEEK(30753)-28672
160 IF C=480 THEN GOSUB 500:GOTO 110
170 GOTO 130
500 PRINT @485,"PRESS ANY KEY TO REPEAT"
510 POKE 29184,96
520 A$=INKEY$
530 A$=INKEY$
540 IF A$="" THEN 530 ELSE RETURN
```

BATTLESHIPS

REPRINTED WITH PERMISSION OF GORDON BROWELL

```
0 ' MICRO MAGIC PROGRAM
1 ' ***"BATTLESHIPS"***
2 'NOT USED IN MICRO MAGIC
10 CLEAR2000:CLS:COLOR,1:PRINT@170,"BATTLESHIPS";:GOSUB1000
20 N$=INKEY$:N$=""
30 PRINT@418,"FOR INSTRUCTIONS (Y OR N)"
50 N$=INKEY$:IFN$=""THEN50
60 IFN$="N"THEN105
70 IFN$="Y"THEN80
80 CLS:PRINT"THE COMPUTER IS CONTROLLING TEN  SHIPS:":PRINT
81 PRINT"   1 BATTLESHIP","   2 CRUISERS"
82 PRINT"   3 DESTROYERS","   4 SUBMARINES"
84 PRINT:PRINT"THEY ARE OF DIFFERENT SIZES -
85 PRINT"      BBBB = BATTLESHIP
86 PRINT"      CCC  = CRUISERS
87 PRINT"      DD   = DESTROYERS
88 PRINT"      S    = SUBMARINES
89 PRINT:PRINT"THEIR POSITIONS ARE KNOWN ONLY   TO WEEZED";
90 PRINT" HEADQUARTERS.":GOSUB1100:GOSUB1100
91 CLS:PRINT@100,"YOUR TASK?"
92 PRINT@258,"...TO SINK THE WEEZED FLEET!":GOSUB1100
93 CLS:PRINT"YOUR SHOTS ARE ORDERED BY GIVING TWO NUMBERS."
94 PRINT:PRINT"THE FIRST NUMBER IS THE ROW,";
95 PRINT:PRINT"      THE SECOND IS THE COLUMN."
96 PRINT:PRINT"A MISS IS SIGNIFIED BY A STAR *"
97 PRINT:PRINT"YOUR SCORE WILL BE SHOWN ALONG"
98 PRINT" WITH YOUR BEST PREVIOUS SCORE."
99 PRINT:PRINT:PRINT"   G O O D   H U N T I N G !!!"
100 GOSUB1100:GOSUB1100:SOUND30,9:CLS
105 X=0
110 A=100
115 DIMG(100)
120 D=0
125 CLS:PRINT@196,"STANDBY - ARRANGING FLEET"
130 C=0
140 FORB=1TO100
150 G(B)=0
160 NEXT
170 E=4
180 F=1
190 H=INT(RND(X)*2)
195 W=0
200 IFH=0THENJ=RND(9)
202 IFH=1THENJ=RND(4)
204 IFH=1THENK=RND(9)
206 IFH=0THENK=RND(4)
220 L=0:P=10*J+K
250 FORM=0TO(E-1)
255 IFH=0THENR=P+M
260 IFH=1THENR=P+10*M
280 IFL=0ANDG(R)<>0THENW=W+1
290 IFL=1THENG(R)=E
300 NEXT
305 IFW>0ANDW<10THEN190
306 IFW=10THEN100
310 IFL=1THEN400
320 L=1
330 GOTO250
```

```
400 F=F+1
410 IFF<4THENE=3
420 IFF>3ANDF<7THENE=2
430 IFF>6THENE=1
440 IFF=11THEN700
445 GOTO190
450 PRINT@435,"     ";:INPUTS
458 IFS<11THEN450
460 IFS>99THEN450
465 T=INT((S)/10)
470 U=S-T*10
472 IFG(S)=5THEN450
475 IFG(S)=4THENS1$="▚"
480 IFG(S)=3THENS1$="▙"
485 IFG(S)=2THENS1$="▜"
490 IFG(S)=1THENS1$="▛"
495 IFG(S)=0THENS1$="*"
500 V=U*2+T*32+101
510 PRINT@V,S1$:SOUND1,6
520 C=C+1
530 IFG(S)>0THEND=D+1
535 G(S)=5
540 PRINT@418,"SHOTS:";C;
550 IFD<20THEN450
555 SOUND4,4;8,4;11,4;16,8
560 IFA<82THENPRINT@457,"BEST SCORE:"A
570 IFC<ATHENA=C
580 PRINT@482,"ANOTHER GAME? Y=YES  N=NO";
590 N$=INKEY$:IFN$=""THEN590
595 IFN$="Y"THEN120
597 IFN$="N"THENCLS:END
600 IFN$<>"Y"THEN580
610 IFN$<>"N"THEN580
700 CLS:PRINT@39,"***BATTLESHIPS***"
720 PRINTTAB(7)"1 2 3 4 5 6 7 8 9":PRINT
730 FORN=1TO9
740 PRINTTAB(4);N;". . . . . ."
750 NEXT
760 GOTO450
1000 FORI=1TO2000:NEXT:SOUND25,6:RETURN
1100 FORI=1TO5000:NEXT:SOUND10,2:RETURN
```

FOR SALE

ORIGINAL VZ PROGRAM TAPES
AND SMALL RANGE P.C.B,S

REASONABLE RANGE OF VZ TAPES AT AUST.$5.00 EACH
SMALL RANGE OF P.C.B'S ALSO AVAILABLE (P.O.A)

SEND SELF ADDRESSED/STAMPED ENVELOPE FOR
A LIST OF TAPES AND P.C.B'S TO:-

DAVE BOYCE
41 HEATHER DRIVE
CHRISTIE DOWNS 5146
SOUTH AUSTRALIA

PLEASE INCLUDE FULL NAME AND ADDRESS

IMPROVING VZ 200/300 SOUND QUALITY :-

The reason for the tinny sound in the VZ is the PIEZO TRANSDUCER. It's located on the bottom of the case under the P.C.B., and because it's glued down it further reduces the sound output. This modification can be done by the beginner.

Before starting the mod you'll have to purchase one of the two sound transducers or 2"" speaker described below and or a miniature pot.

1) 5.1cm (2") 8 Ohm speaker. TANDY Cat. No. 40-245 - $3.99
2) PIEZO TRANSDUCER 1 - 30mm X 5mm - Available from D.Smith.
3) PIEZO TRANSDUCER 2 - 30mm X 10mm - TANDY No. 273-073 - $2.99
4) MINIATURE ROTARY POT (5K) - Available from D.SMITH - $1.20

I tried all three and found the 30mm X 10mm P/TRANSDUCER the most satisfactory as far as sound output, size and price. Sound output between the 5.1cm speaker and 30mm X 10mm P/TRANSDUCER was nearly identical. So size and price took priority.

The P/TRANSDUCER element used in the VZ is the same as the two described previously. The one in the VZ is uncased, while the other two are cased. the larger the case the more sound output. And now to the mod.

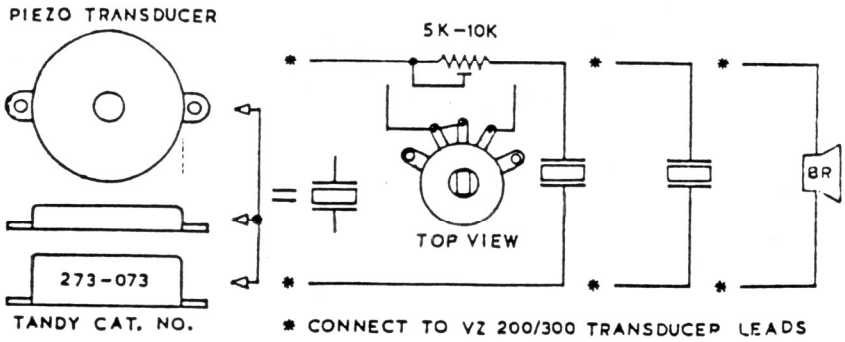
Turn your VZ upside down and remove all the screws holding the two halves of the case together and turn VZ right way up and back to front. Lift top of case and put towards the back.

The screws holding the P.C.B. in place will have to be removed next. Lift the P.C.B. out carefully and you should see the P/TRANSDUCER element glued to bottom of case. Unsolder or cut from the element the two leads going to it.

If you desire a volume control, now is the time to insert it in one of the leads as shown in circuit diagram. It can be mounted on bottom left lip of the case. Drill a hole to suit the shaft and mount it upside down with the knob going under the lip. The volume control is linear in operation and at mimimum is like the volume from original VZ while at Maximum is fair bit louder.

Next solder the leads from your new TRANSDUCER to the ends of the unsoldered leads. If you have'nt a soldering iron then simply twist the wires together. Make sure you insulate the joined wires from each other and the VZ by using sleeving or wrapping some electrical tape around the joints.

The best place to put the P/TRANSDUCER is at front left of case with the leads going under the P.C.B. Use a bit of tape on one side of P/TRANSDUCER to keep it in place. All that's left now is to reasemble. I think you'll agree, it was'nt hard.



NEWSLETTER INDEX
PART 2-ISSUE 16 TO 20

ISSUE 16..

PROGRAM-VSWR CALCULATOR (KEITH WRIGHT)
REVIEW-LASERLINKS 'EXTENDED BASIC' (PETER PATTERSON)
WRITE PROTECT OVERRIDE SWITCH (JOE LEON)
PROGRAM-SNOOPY CALENDAR-PART 2 (DAVE BOYCE)
CHANGES TO SNOOPY CALENDAR FOR PRINTER PLOTTER (DAVE BOYCE)
ASSEMBLY LANGUAGE-PART 1 (MARK HARWOOD)
UNDERSTANDING YOUR VZ-PART 5 (ROBERT QUINN)

ISSUE 17..

UNDERSTANDING YOUR VZ-PART 6
REVIEW-EXT.DOS V1.0
ASSEMBLY LANGUAGE-PART 2
PROGRAM-TWISTER (RICHARD DAVIS)
PROGRAM-LOAN (KEITH WRIGHT)
AEM 4505 SPEECH SYNTHESISER- TONE CONTROL (DAVE BOYCE)
PROGRAM-HAM CONTACT LOG FROM MICRO MAGIC BY GORDON BROWELL

ISSUE 18..

REVIEW-GALACTIC EMPIRES (P HILL)
INTRODUCTION TO PROGRAMMING-PART 1 (BOB KITCH)
ASSEMBLY LANGUAGE-PART 3
AEM 4505 SPEECH SYNTH-PROGRAMS (DAVE BOYCE)
INKEY$ INPUT ROUTINE (PAUL LEON)
PROGRAM-SIDEWAYS (DAVE BOYCE)
PROGRAM-NZ TWISTER AGAIN (RICHARD DAVIS)
BOOK REVIEW-ASSEMBLY LANGUAGE PROGRAMMING MANUAL (ROSS WOODS)

ISSUE 19..

INTRODUCTION TO PROGRAMMING-PART 2 (BOB KITCH)
ASSEMBLY LANGUAGE-PART 4
POKING AROUND WITH THE VZ (P HILL)
REVIEW-PRINTER PATCH V1.4 FROM HUNTER VALLEY VZ JOURNAL
PROGRAM-WORLD TIMES (RICHARD DAVIS)
REVIEW-MONOPOLY (P HILL)
SHIFT LOCK SWITCHES (JOE LEON)

ISSUE 20..

PROGRAM-ELEMENTS AND SYMBOLS (P LEON)
STARTER HINTS FOR BEGINNERS (J D'ALTON)
DOT MATRIX GRAPHICS (LARRY TAYLOR)
PROGRAM-DOT MATRIX GRAPHICS EDITOR (LARRY TAYLOR)
NZ TWISTER CORRECTION
GRAPHICS EXAMPLE (L TAYLOR)
ASSEMBLY LANGUAGE-PART 5
PROGRAM-HI RES CHARACTERS (RICHARD DAVIS)
LIST OF COMMERCIAL SOFTWARE FOR VZ (EDDIE TOMES)
PROGRAM-SPOT LIGHT (DANNY TODD)

SOFTWARE FOR SALE

We now have the following software available for sale.
All prices shown include postage and packing anywhere in New Zealand.
Please make sure that the memory of your VZ is greater than or equal to the memory required by the software that you wish to purchase.
Memory sizes indicated do not include video ram.

ALL PRICES ARE SHOWN IN NEW ZEALAND DOLLARS.

BLACKJACK ROYALE	16K	$15.00	HAUNTED MANSION	22K	$15.00
CASTLE GREYSTONE	32K	$15.00	KNIGHTS QUEST	32K	$18.00
DATA	16K	$18.00	NAMESTAR/SHOPSTAR	22K	$15.00
FBI-2001	31K	$18.00	SCOTLAND YARD	16K	$ 8.00
GALACTIC EMPIRES	16K	$15.00	TRIVIAL CULT	16K	$ 8.00
GRAPHSTAR	22K	$24.00	VZ MONOPOLY V2.0	22K	$18.00

PLEASE MAKE ALL CHEQUES AND MONEY ORDERS PAYABLE TO-

AUCKLAND VZ 300/200 COMPUTER CLUB.

VZ DISASSEMBLER FOR SALE

This Disassembler is written by Peter Hickman.
And is also distributed as the "Laserlink Disassembler".

Peter is offering this very fine program to user group members for NZ$24.00.
All orders can be placed through this club,as peter has offered a cheaper price for bulk purchases.

The program is available on tape or disk.And comes in a 4 program version and
You choose which section you use depending on what memory you have.

ALL THESE PROGRAMS CAN BE DEMONSTRATED AT THE WORKSHOP IF REQUESTED

P.O.BOX 1972 C.P.O AUCKLAND N.Z

(C) AUCKLAND VZ 300/200 COMPUTER USER'S CLUB 1986

ALL MATERIAL IN THIS NEWSLETTER IS SUBJECT TO COPYRIGHT..
CONTRIBUTED PROGRAMS & ARTICLES CAN BE REPRODUCED ON THE UNDERSTANDING THAT THEY ARE FOR THE PRIVATE USE OF SUBSCRIBERS ONLY.
COPYRIGHT IS ALWAY'S RETAINED BY THE AUTHOR.
THIS CLUB WILL NOT BE HELD RESPONSIBLE FOR THE BREECHING OF COPYRIGHT CAUSED BY THE PUBLICATION OF ANY MATERIAL SENT IN UNDER A FALSE NAME.

YOU ARE WELCOME TO PHONE ME BETWEEN 6.30 PM AND 9 PM ----PHONE 673-967